uProcessWatch: Monitoring MOOS Application Health

June 2018

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

1 Overview

The uProcessWatch application monitors the health of a set of MOOS application. It does two things:

- It monitors the presence of a set of MOOS apps,
- It monitors the CPU load of a set of MOOS apps.

In the case of the former, uProcessWatch continually monitors the DB_CLIENTS list for applications it has responsibility for watching. In the case of the latter, MOOS apps are already monitoring and reporting their own CPU load and uProcessWatch is doing nothing more than gathering that information for display in the terminal or appcast message. An example terminal output is shown below:

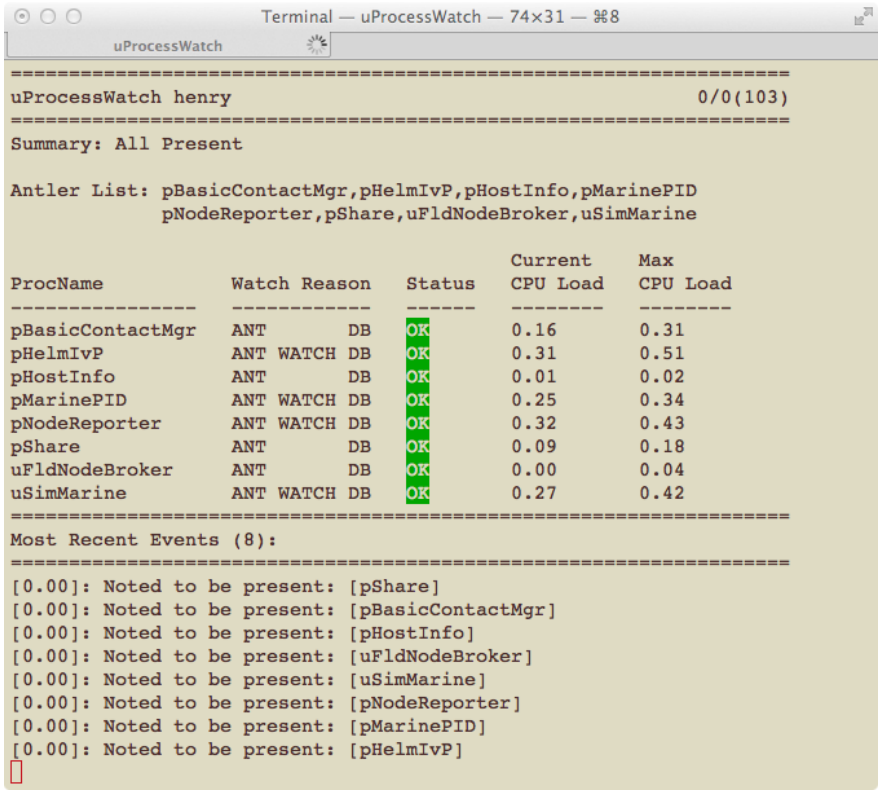

Figure 1: **A typical terminal or appcast report for `uProcessWatch`.**

The first line, "Summary: All Present", tells you that no processes are missing. This is also posted in the variable `PROC_WATCH_SUMMARY`. The list of applications launched with `pAntler` is shown in the next block. The main body shows, for each watched application, the reason why the process is on the watch list, the status, current CPU load as reported by the process itself, and the maximum CPU load noted so far.

The bottom section of the terminal output shows the events (similar to all appcasting output). In the case of `uProcessWatch`, events note either the arrival or disappearance of a watched process. Each event also results in the posting of the variable `PROC_WATCH_EVENT`. The user may configure `uProcessWatch` to *not* watch certain named applications or patterns of applications, such as `uXMS*`, to avoid unwarranted alerts.

2 Typical uProcessWatch Usage Scenarios

2.1 Using uProcessWatch with AppCasting and pMarineViewer

The first usage scenario is perhaps the most typical since it is viable both in simulation and in the field where the vehicles have a network connection to a shoreside MOOS community. The idea is shown in Figure 2 below. Multiple vehicles each run `uProcessWatch` locally. A network connection from each vehicle to a shoreside MOOS community is used to share appcast messages using `pShare`. The shoreside community is running `pMarineViewer` which supports a multi-vehicle appcast viewing

mode. In this way, the shoreside user may monitor the health of all vehicles' applications with one tool. If a process on a single vehicle goes missing, the menu item on the shoreside viewer for that vehicle turns red.

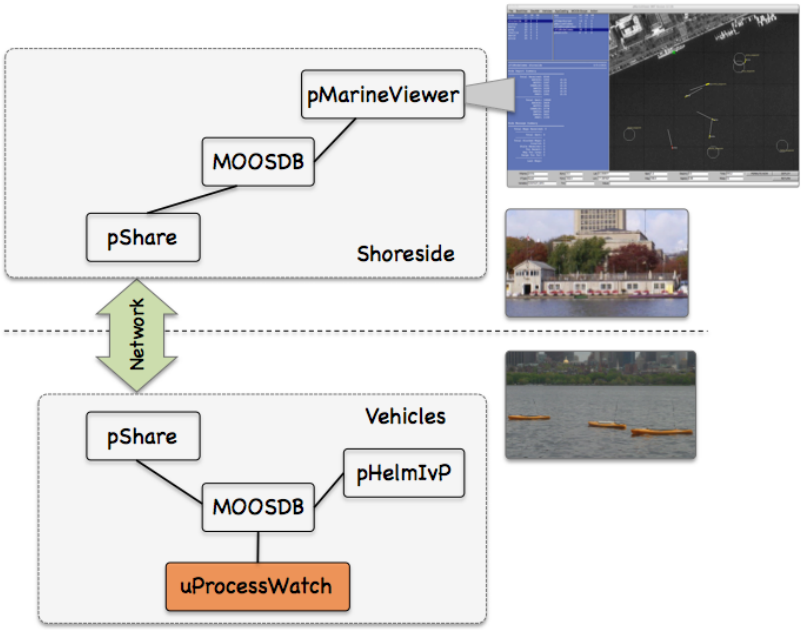

Figure 2: **Using uProcessWatch with AppCasting:** uProcessWatch is run locally on each fielded vehicle, generating appcasts posted to the local MOOSDB. Appcasts are shared to the shoreside and collected by the pMarineViewer tool for monitoring processes across all vehicle. This scenario is relevant when there is a network connection from the vehicles to the shoreside.

2.2 Directly Accessing the PROC_WATCH_SUMMARY Output

The main health indicator produced by uProcessWatch in the MOOS variable PROC_WATCH_SUMMRY. This can be checked on a remote machine by simply scoping on this variable. The uMS application distributed with MOOS will do the trick for example. To focus solely on this variable, the uXMS tool may also be used as follows:

```
$ uXMS --server_host=10.25.0.72 --server_port=9000 PROC_WATCH_SUMMARY
```

Or one may ssh onto the vehicle and launch a scope locally on this variable.

3 Using and Configuring the uProcessWatch Utility

The primary configuration of uProcessWatch is defining the *watch list*, the set of other MOOS processes to monitor. By default all processes ever noted to be connected to the MOOSDB are on the watch list. The same with all processes name in the Antler configuration block of the mission file. This default is used because it is simple, and a good rule of thumb is that if any process disconnects from the MOOSDB, it's probably a sign of trouble.

3.1 The DB_CLIENTS Variable for Detecting Missing Processes

The MOOSDB, about once per second, posts the variable DB_CLIENTS containing a comma-separated list of clients (other MOOS apps) currently connected to the MOOSDB. When uProcessWatch is configured to watch for all processes, it simply augments the watch list for any process that ever appears on the incoming DB_CLIENTS mail. If the process is then missing at a later reading of this DB_CLIENTS mail, is is considered AWOL.

3.2 Defining the Watch List

The *watch list* is a list of processes, i.e., MOOS apps. Each item on the list will be reported missing if it does not appear in the list of clients shown by the current value of DB_CLIENTS. By default, all clients that ever appear on the DB_CLIENTS list will be added to the watch list. The same is true for all processes named in the Antler configuration block of the mission file. This default can be overridden by the following configuration option:

```
watch_all = false
```

The value of watch_all may also be set to "antler" to indicate that items on the Antler list are to be watched by default, but not those that appear in the DB_CLIENTS list. Likewise it may be set to "dbclients" to indicate that items in the DB_CLIENTS list are to be watched by default, but not those in the Antler list. Processes may be added to the watch list by explicitly naming them in configuration block as follows:

```
watch = process_name[*]
```

A process may be named explicitly, or the prefix of the process may be given, e.g., watch=uXMS*. This will match all processes fitting this pattern, e.g., uXMS_845, uXMS_23.

Processes may be explicitly *excluded* from the watch list with configurations of the following:

```
nowatch = process_name[*]
```

This is perhaps most appropriate when coupled with watch_all=true. If a process is for some reason both explicitly included and excluded, the inclusion takes precedent.

3.3 Reports Generated

There are three reports generated in the variables:

- PROC_WATCH_EVENT
- PROC_WATCH_SUMMARY,
- PROC_WATCH_FULL_SUMMARY

All reports are generated only when there is a change of status in one of the watched processes. The first type of report is generated for each event when a watched process is noted to have connected or disconnected to the MOOSDB. The following are examples:

```
PROC_WATCH_EVENT = "Process [pMarinePID] is noted to be present."
PROC_WATCH_EVENT = "Process [pMarinePID] has died!!!!"}
PROC_WATCH_EVENT = "Process [pMarinePID] is resurected!!!"
```

In the first line above, a process is reported to be present that was never previously on the watch list. In the third line a process that was previously noted to have left the watch list is reported to have returned or been restarted.

The PROC_WATCH_SUMMARY variable will list the set of processes missing from the watch list or report "All Present" if no items are missing. For example:

```
PROC_WATCH_SUMMARY = "All Present"
PROC_WATCH_SUMMARY = "AWOL: pMarinePID,uSimMarine"
```

The PROC_WATCH_FULL_SUMMARY variable will list a more complete and historical status for all processes on the watch list. For example:

```
PROC_WATCH_FULL_SUMMARY = "pHelmIvP(1/0),uSimMarine(1/1),pMarinePID(2/2)"
```

The numbers in parentheses indicate how many times the process has been noted to connect to the MOOSDB over the number of times it has been noted to have disconnected. A report of "(1/0)" is the healthiest of possible reports, meaning it has connected once and has never disconnected.

3.4 Watching and Reporting on a Single MOOS Process

If desired, uProcessWatch may be configured to generate a report dedicated a single MOOS process with the following example configuration:

```
watch = pBasicContactMgr : BCM_OK
```

In this case a MOOS variable, BCM_OK, will be set to either true or false depending on whether the process pBasicContactMgr presently appears on the list of connected clients in listed in DB_CLIENTS.

3.5 A Heartbeat for the Watch Dog

By default uProcessWatch will only post PROC_WATCH_SUMMARY when its value changes. A long stale PROC_WATCH_SUMMARY = "All Present" likely means that everything is fine. It could also mean the uProcWatchSummary process itself died without ever posting anything to suggest otherwise. The user has the configuration option to post PROC_WATCH_SUMMARY every N seconds regardless of whether or not the value has changed:

```
summary_wait = 30  // Summary posted at least every 30 seconds
```

This in effect creates a heartbeat for monitoring uProcessWatch. The default value for this parameter is -1. Any negative value will be interpreted as a request for postings to be made only when the posting value changes. Regardless of the summary_wait setting, the other two reports, PROC_WATCH_EVENT and PROC_WATCH_FULL_SUMMARY, will only be made when the values change.

3.6 Excusing a Process

An application on the watch list may be excused and disconnect from the MOOSDB if it posts to the variable EXITED_NORMALLY with the name of itself. A check is made by uProcessWatch of the *source* of the posting to ensure that message was posted by the exiting process. It's up to the developer of an application to build in the feature declaring a normal exit. An example is the uTimerScript application which has the ability to execute a script of postings to the MOOSDB followed by an exit. In this case, the *status* of application becomes EXCUSED, as shown in Figure 3.

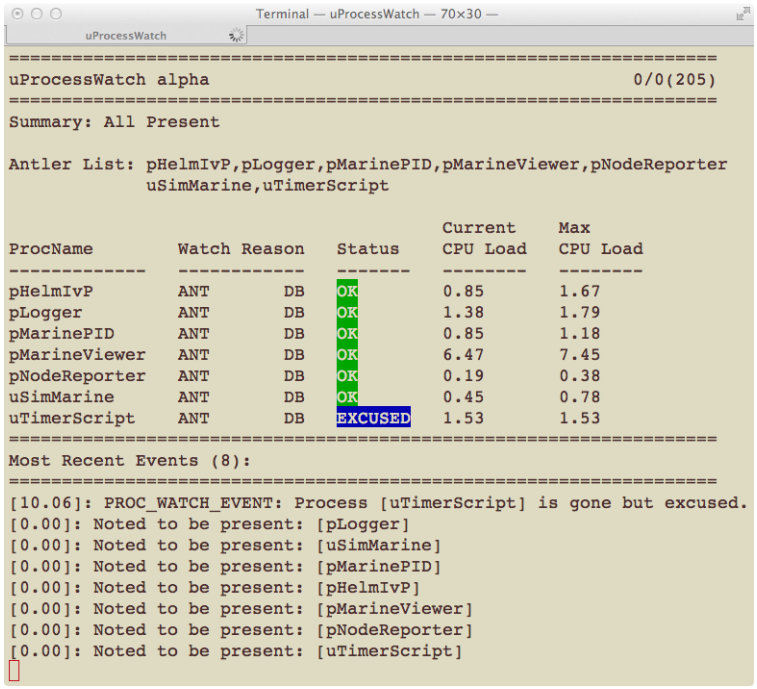

Figure 3: The process uTimerScript has gone missing due to its own intentional exit. Before exiting it posted EXITED_NORMALLY=uTimerScript, and therefore uTimerScript is regarded as excused.

Excusing a missing process is different than choosing to not include it on the watch list. Some applications are, by their nature, designed to disappear at some point. Scoping tools like uXMS, uPokeDB or uHelmScope are examples applications that regularly appear and disappear. They are best excluded entirely from the watch list with the nowatch configuration parameter.

3.7 Allowing Retractions if a Process Reappears

With the addition of appcasting to uProcessWatch, a missing process also triggers an appcast run warning, as shown on the top in Figure 4 below.

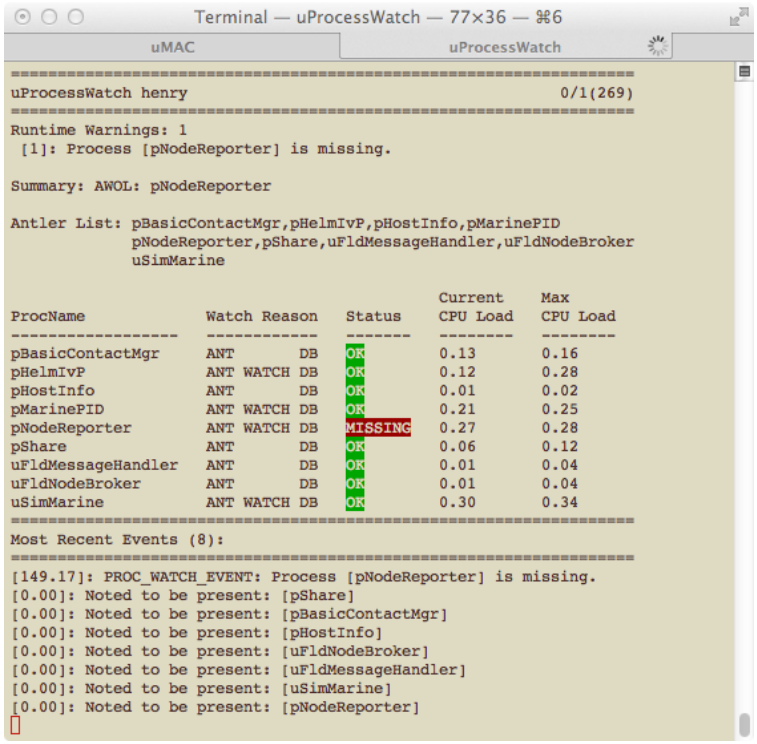

Figure 4: The process pNodeReporter has gone missing. This is noted as a runtime warning at the top, and the MISSING status in the body of the report.

However, there are cases where a process goes missing but then returns, in which case the warning is a distraction. These reasons include.

- The application may actually exit and then restart a short time later.
- The application my be running at a very slow apptick in which case it may be dropped from the DB_CLIENTS list momentarily.
- The application may be missing only because it hasn't launched yet.

To address this, uProcessWatch will allow retractions on appcast run warnings. If the application disappears and reappears, the appcast run warning will disappear. The successive posted values of PROC_WATCH_SUMMARY will show that the process was at some point declared AWOL, but once the process returns, the appcast output will no longer raise attention to the issue.

Of course there are situations where a process that disappears and then reappears is an indication of a problem, not to be swept under the rug. In this case the default behavior of uProcessWatch may be changed by setting allow_retractions to false. Presently this setting cannot be set on a per-process manner.

4 Configuration Parameters of uProcessWatch

The following parameters are defined for uProcessWatch. A more detailed description is provided in other parts of this section. Parameters having default values indicate so in parentheses below.

Listing 4.1: Configuration Parameters for uProcessWatch.

allow_retractions:	If true, run warnings are retracted if a process reappears after disappearing. Legal values: true, false. The default is true.
nowatch:	A process or list of MOOS processes to *not* watch.
post_mapping:	A mapping from one posting variable name to another.
summary_wait:	A maximum amount of time between PROC_WATCH_SUMMARY postings. Negative value indicates posting occurs only when the value changes regardless of elapsed time. Legal values: any numerical value. The default is −1.
watch:	One or more comma-separated MOOS process to watch and report on.
watch_all:	If true, watch all processes that become known either via the DB_CLIENTS list or the Antler list. Legal values: true, false. The default is true.

4.1 An Example MOOS Configuration Block

Listing 2 shows an example MOOS configuration block produced from the following command line invocation:

```
$ uProcessWatch --example or -e
```

Listing 4.2: Example configuration of the uProcessWatch application.

```
ProcessConfig = uProcessWatch
{
  AppTick   = 4
  CommsTick = 4

  watch_all = true   // The default is true.

  watch  = pMarinePID:PID_OK
  watch  = uSimMarine:USM_OK

  nowatch = uXMS*

  allow_retractions = true   // Always allow run-warnings to be
                             // retracted if proc re-appears

  // A negative value means summary only when status changes.
  summary_wait = 10 // Seconds. Default is -1.

  post_mapping = PROC_WATCH_FULL_SUMMARY, UPW_FULL_SUMMARY
}
```

5 Publications and Subscriptions for uProcessWatch

The interface for uProcessWatch, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ uProcessWatch --interface or -i
```

5.1 Variables Published by uProcessWatch

The primary output of uProcessWatch to the MOOSDB is a summary indicating whether or not certain other processes (MOOS apps) are presently connected.

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility.
- PROC_WATCH_EVENT: A report indicating a particular process has been noted to be gone missing or noted to have (re)joined the list of active processes.
- PROC_WATCH_FULL_SUMMARY: A single string report for each process indicating how many times it has connected and disconnected from the MOOSDB.
- PROC_WATCH_SUMMARY: A report listing all missing processes, or "All Present" if no processes are missing.

The user may also configure uProcessWatch to make a posting dedicated to a particular watched process. For example, with the configuration watch=pNodeReporter:PNR_OK, the status of this process is conveyed in the MOOS variable PNR_OK, set to either true or false depending on whether or not it is present.

The variable name for any posted variable may be changed to a different name with the post_mapping configuration parameter. For example, post_mapping=PROC_WATCH_EVENT, UPW_EVENT will result in events being posted under the UPW_EVENT variable rather than PROC_WATCH_EVENT variable.

5.2 MOOS Variables Subscribed for by uProcessWatch

The following variable(s) will be subscribed for by uProcessWatch:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- DB_CLIENTS: A comma-separated list of clients currently connected to the MOOSDB, posted by the MOOSDB. Used for detecting missing processes. Section 3.1.
- EXITED_NORMALLY: An indication made by a process, potentially on the watch list, that it has exited normally and should be excused, and not regarded as missing. Section 3.6.
- <PROCNAME>_STATUS: The status string generated for all MOOSApps, containing the current CPU load among other things.